**Notes**

**Room Db Creation**

1. Create entity class that represents a table, Then create entity columns with it’s getters and setters
2. Create Dao Class ”Data access objects ” with crud operations, any custom query annotate it as query and write the query
3. Create RoomDatabase
   * It must be abstract class
   * It must extend RoomDatabase
   * It must be a singletone
   * Annotate it with database that takes
     + array of entities
     + Version number that must be increased on any update to database structure
4. Synchronized means one thread at a time can access this singleton object in case of multithreading
5. Repository class provide an abstraction layer between data layer and rest of application it provides data from multiple sources
   * We create the database in it’s constrctor and init dao object and get all notes liveData List
   * Then we provide methods for all db opreations and do them in another thread for example async tasks to prevent app from freezing
   * All async tasks must be statics if they are inner classes to prevent memory leaks
6. To insert some data on creating the room database we
   * Create static callback method
   * It’s return type is RoomDatabase.Callback
   * And make a new object and override oncreate method
   * Use async task to add to database and send instance as the db dao object
7. Creating A viewModel , it’s aware of activity lifecycle and responds to it
   * First extends androidViewModel not ViewModel cause AndroidViewModel Takes an application as context which will be useful to send it to repository then use it to init database
     + but on other hand if we use ViewModel we should then send activity context and if we holed a reference to an activity it’ll cause a memory leak
     + , cause viewmodel is designed to outleft the activity after it’s destroyed then we will have a refreance to destroyed activity
   * Create references to the “Repository” Class and another one to the “liveDataList” and init them in the constructor
   * Then, our activity has no access to the repository class so we create methods in the viewmodel to do repository opreations(insert,update,delete…etc)
8. Create refrance to the view model in our activity
   * We don’t use new
   * We ask android view model providers cause it knows when to create new one and when to pass it directly
   * View model provider takes
     + owner to inform the viewModel which lifecycle to follow
     + and the viewModelClass that we want object from
9. When we get a LiveData of Database entity we can set observer to it,
   * It takes the owner which is the activity cause livedata is lifecycle aware and it follows the owner
   * Then the observer class to listen to changes
   * In the onChange we can update ui the liveData will update if the activity in the foreground and not when the activity is destroyed
10. Add note Activity we can simply create new object of our view model in this activity but then we will have a list of notes and that is not it’s responsibility to have list of notes so:
    * Better we create more than view model classes and distribute the operations
    * But in our example we will make add note activity returns a result with the data user entered and use it in the main activity onresult method and use our viewmodel to insert that note to the db
11. Delete note on swipe left or right
    * Create item touch helper
    * Pass swipe dirctions to it’s constructor
    * Onswipe method get the note at the position swiped
    * Call view model delete note
12. To make on item click in recycler view
    * First we make an interface inside the adapter
    * Contains ”onITemClick “abstract method and takes an object of note model
    * Then make object of that interface inside the adapter and make a setter method to it
    * Then in the constructor of the view holder
      + Set on itemview click listener
      + and call listener.onitemclicklistner(ITem);
    * And pass the object of the current position
13. To make animation to the reyclerview on add ,edit or delete we do he following:
    * The problem is that we don’t know at which position the changes happened
    * There is a class called DiffUtls helps us to compare two lists
    * There is a class called ListAdapter that implants that technique for adapters and do most of the code for us and the best thing is that it makes all the comparison in a background thread so app won’t freeze or crash
    * So make our adapter extends ListAdapter and pass to it
      + Note
      + ViewHolder
    * Make constructor that matches the supe the on with DiffCallback In it
    * Make a static diffCallBack and call super(diffcallBack) instead of taking it as parameter to the constructor
    * In AreItemsTheSame we retrun true if the old item and the new item are the same object but may contents changed or not so we check with the item id
    * In AreContntsTheSame we return true if no changes happened to the old object and return false if item contents changed
      + We do this by comparing each field in our old item object with the new item object
    * Then we will no longer need to have object of our list in the adapter so remove it and make the proper changes
      + A default method to get item at position getItemAt(position)
      + Remove getcount() the list adapter will take care of that method
      + Delete setNotes() and we will replace it with another default method in the listAdapter Class “submitList()” and call it from main activity